2009 Fifth International Conference on Autonomic and Autonomous Systems

Heterogeneous QoS Resource Manager with Prediction

Ramon Nou [1], Jordi Torres [2]

Barcelona Supercomputing Center (BSC), Technical University of Catalonia (UPC)
Barcelona (Spain)
[1]rnou@ac.upc.edu [2]torres@ac.upc.edu

Abstract

As long as computers continue to get more CPU processing power, data centers need to optimize their power usage. We can do this and maintain the same complexity level as before by using virtualized environments. We can put a large number of small isolated servers, inside a large one and improve a large number of values like the wattage or power consumption, space usage, and resource usage. In this paper, we present a prototype with which we distribute resources between two virtualized servers, one with Tomcat and another with Globus, and both sharing the same host. The prototype is able to maintain the required SLA and QoS using prediction with simulation in real time. Our goal is to demonstrate that simulation can be used to improve resource managers decissions. In this paper, we use those simulations inside a shared server with several different applications using virtualization.

1 Introduction

Increasing the power of computers improves performance, but in a lot of situations also increases the idle time of a server. This fact has some serious backdraws, firstly we needlessly spend energy (powering up and cooling the server), secondly we use space (racks, power outlets) and thirdly, we waste time and effort on maintenance. Although there are efforts to produce energy-efficient servers, our analysis using a normal server (8-way Pentium Xeon 2.6 GHz with the latest energy-aware kernel (2.6.23 and patches) shows only 160 watts of difference between a 100% used server and an idle one. An idle server, like that previously described, uses 420 watts, whereas a 100% used server needs 580 watts as shown in figure 1 so it makes a lot of sense to fully use a server.

With these results, they show that sharing a server is a must. Virtualization reduces the cost and greatly improves the management of sharing servers and resources. It allows

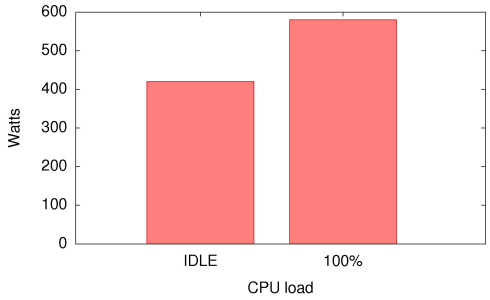

Figure 1. Wattage difference between an idle server and a 100% used server.

for the same physical resources to be used in different services while maintaining isolation, however, we still need a resource manager to dynamically adapt the resources to the application's needs (as seen on [15]).

In this paper, we will show the creation of an autonomic heterogeneous environment based on our previous work. The environment is similar to that in the work presented at [15] but now includes a virtualized environment using Xen [5], a set of unmodified applications, and prediction using our previous simulations (using OMNeT++ and QPN (SimQPN)). The resource manager decides, with the aid of those simulations, how to share processors using the following two SLA (Service Level Agreement) policies:

1. Apply an SLA on the throughput level to Tomcat.
2. Apply an SLA for the response time of the sessions willing to run in Globus.

In this paper, for the sake of simplicity, we decided to give priority to Tomcat (first SLA policy) over Globus. If needed we could assign them the other way around or even apply a third priority SLA. With our actual policy, Tomcat will obtain the maximum performance, always following its SLA, and Globus will reduce its workload (canceling

978-0-7695-3584-5/09 $25.00 © 2009 IEEE
DOI 10.1109/ICAS.2009.25

sessions, or reducing the level of parallelism) while maintaining the QoS with the unused resources that the server has.

As related work we have [18] where a data center with three applications (all transactional) are sharing resources. The resource manager uses control theory with a black-box taking decisions. Similar work appears in [7] which provides dynamic, web only, application placement on a cluster and adjusts resources according to its resource demand. Its goal is to maximize the utilization of the servers. Our work presents a heterogeneous workload and the use of simulation results in more flexibility in some cases.

The rest of the paper is organized as follows. In Section 2, we introduce our resource manager architecture and the different simulators. In Section 3, we describe the experimental environment in which we evaluate our framework. Section 4 presents the detailed results of our evaluation. Finally, in Section 5, we present some concluding remarks.

2 Design

A system diagram is shown in figure 2. We have a set of six workload generators. They generate requests to the virtualized servers (*Tomcat VM* and *Globus VM*). A Tomcat workload and SLA analyzer captures the workload characteristics of the client requests. Actually it is a known workload based on the RUBiS [1] Benchmark, but we could put an automatic learning block here. When the system detects a change in the workload or in the SLA, we run a set of simulations to select the best assignment of resources that ensure the SLA level is met (using the algorithm shown below) and finally, it informs the *VM Resource Manager* (VMRM) of the decision. It can be seen that the VMRM has a back arrow to the Tomcat Analyzer; we don't actually use it for this experiment but it should be used to inform about the final decision on the resource assignation, if it changes. On the Globus side, we have a *Globus QoS Broker* (details in [10] and [16]) that decides to reject, reduce the parallelism of, or do session rescheduling on the Globus session request that arrives in the system. Its decisions are taken using the Globus Simulator with the known number of CPUs available as an input parameter. Finally, when a change in the assignation is needed, VMRM speaks to Xen (in the Domain-0) and does a change of the virtual machines' resources.

```
if (|oldSLA − newSLA| < ε ∩
   |oldRequest − newRequest| < ε ∩
   |oldClientRate − nCR| < ε) do
begin
 CPU := 0
 found := false
 replies := ∅
 while (CPU ≤ availableCPU ∪ ¬ found) do
 begin
   CPU := CPU + 1
   replies[CPU] := Sim (CPU, newRequest, nCR)
   found := replies[CPU] ≥ newSLA
 end
 if (found) do
    XenAssign (Tomcat, CPU)
 if (¬ found) do
 begin
  CPU := availableCPU
  diff := 0
  while (diff < ε ∪ CPU > 1 ) do
  begin
   diff := |replies[CPU] − replies[CPU − 1]|
   CPU := CPU − 1
  end
  if (diff < ε ) do
     XenAssign (Tomcat, CPU)
  else do
     XenAssign (Tomcat, CPU + 1)
 end
end
```

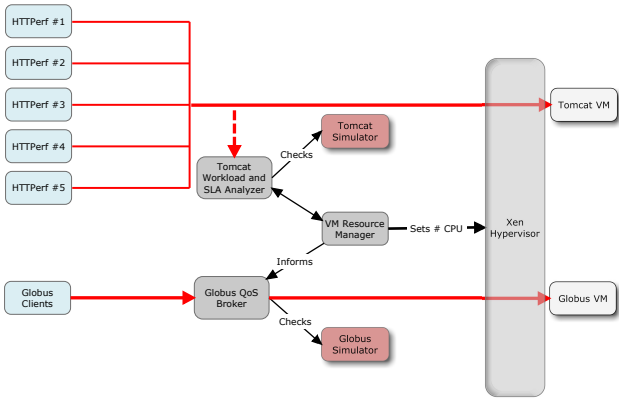

Figure 2. Prototype design with Tomcat priority over Globus assigned.

Note that the Resource Manager only informs the Globus QoS Broker of the number of resources available, whereas the VMRM informs and gets the result of the prediction and the decision of the Tomcat Manager (because it has priority over Globus, as we said before). Although we have shown the diagram for a specific case, it can grow to a large number of applications, and can also use other physical or virtual servers for Globus scheduling for example.

The two simulations were built using OMNeT++ [19]. Further details on the simulations can be found in [10, 14, 17]. There are other similar simulation frameworks like REPAST [13], GridSim [3] (which simulates a Grid cluster with great precision) and SimJava [11] but OMNeT++ was easy to use, and well-known by us. Also, we didn't need many of the features that GridSim provides.

2.1 Tomcat Simulation

The Tomcat simulator is based on our simulation that appeared in [14]. We have modified it in order to make it usable on the new environment. First, since we are using the

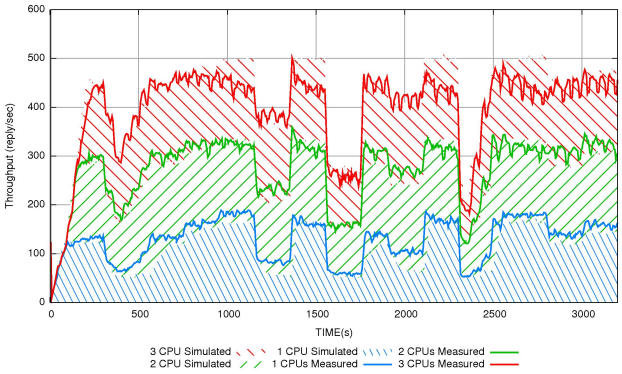

Figure 3. Comparison of the results obtained with simulation (dashed-area), using 1, 2 or 3 processors and a dynamic workload. The lines represents the measured values.

resource "processor", we need to include it in the simulation. We want to simulate the context changes and other behaviors (related to scheduling) that happens on the real server. We also included MySQL inside the server sharing the resources. Finally, we included some mechanisms to be able to stop and continue the simulation (store simulation state), which we will need in more complex scenarios where we need to be aware of the past. In this paper, we use the ability to initialize the simulation in a specific state; in dynamic and continuous environments we need to use the simulation in other states than that of the initial one.

The evaluation of the simulator can be found in previous work, but we had run a comparison between measured vs simulated with a dynamic workload. A dynamic workload evaluation is important as long as we are using the prediction in a real (and continuous) time system.

In figure 3 we can see how the server behaves with a dynamic workload without admission control in a server with 1, 2 or 3 processors. The lines show the actual measured values and the pattern areas represent the values obtained with the simulation of the same environment. Due to these results and the older ones, we can get ensure that the simulations and its parameters are valid to represent this application.

2.2 Globus Simulation

To simulate Globus we will use the simulation created in [10, 17] with some additions like the destruction of servers and addition of new servers. We will use it by defining 3 servers of 1 CPU to simulate the environment, which is a semantic change that allows us to use the same simulation design without changes. The QoS broker will decide, based on the prediction results, if we need to reduce the parallelism of a session or cancel it. The objective is to use all the resources while maintaining the SLA and the QoS.

In the previous work we used a QPN [8] with SimQPN [9] to simulate Globus. Although it produced very good results (Globus is not overloaded, so analytical methods are valid), we created a new simulation with OMNeT++ for this paper to allow us more flexibility. The design is the same as the one found in [17].

3 Experimental Environment

The experimental environment consists of a virtualized environment using Xen 3.1 [2]. We have two servers (Globus and Tomcat) in a 4 way 3.16GHz Pentium Xeon with 10 GB of RAM. The 4 processors (CPU_{Total}) will be assigned dynamically to the two virtual machines satisfying $CPU_{Total} = VCPU_{Globus} + VCPU_{Tomcat}$ and $VCPU_{GLOBUS} > 0 \cup VCPU_{TOMCAT} > 0$. With these rules we didn't have collisions between the domains and the physical processor is identified with a virtual processor. The granularity of the assignation is done at the processor level, but we could also use a smaller sharing unit like half-processors. Using a smaller unit will increase the time to make a decision because we will have a wider search space. More specifically, if we use half CPUs we would require an extra simulation. On the other hand, the benefits of using prediction will increase.

The Globus virtual machine has a default installation of Globus Toolkit 4.0.5 [6]. In the Tomcat virtual machine we have an unmodified Apache Tomcat 6.0.16, with an MySQL database; we have put the database inside for sake of simplicity. The workload level is very high and it saturates the network and the database server. This introduces a small change in the simulator because we need to connect the MySQL server to the same scheduler as that of the Tomcat subsystem as we showed in section 2.

The Globus clients are generated with a unique machine (CPU and network utilization in the client is minimal), but we need at least five machines with HTTPerf [12] and RUBiS workload to generate a high load for Tomcat. Finally, there is a machine with 2 processors that is running the resource manager and the simulations. This machine also works as a proxy server and controls the requests sent to the virtualized servers. The resource manager sends requests to change processor assignation to the Domain-0 of Xen inside the host with the virtualized servers. The system is outlined in the previous figure 2.

4 Evaluation

In this section we will evaluate our proposal. All the evaluation has been done on a real environment. More evaluation of each of the simulators and the QoS Broker can be

found in papers [10,14,16,17]. We decided to use this workload and settings to simplify the plots and explanations, but other workloads and settings (like using more processors or half processors) will increase the benefits of our proposal.

4.1 Resource Manager with Prediction

In figure 4 there are 3 plots. First shown is the workload of the two servers: The line for the Tomcat workload represents a dynamic load (we modify the number of new clients per second that are generated). Every new client follows a request distribution specified by RUBiS. We also have the session requests that the clients send to Globus. Further details can be found in [10]. To simplify, every session request contains a response time SLA, a number of service, the number of service requests and the interarrival time. A Globus session request has a mean lifetime of 80 service requests. Although we have simplified the SLA to only consider the response time, we can use more complex attributes, for example, WSLA [4] and provide values such as penalties if an SLA is broken.

To analyze the results, we should take note of the behavior of RUBiS/HTTPerf: In our setup a client is maintained in the system, doing requests, for nearly 1900 seconds. Although the workload is low or stable in value, the requests are accumulated, increasing the throughput if the CPU power is enough.

The second plot shows the SLA of the Tomcat Server. Anyone who rents the server requires a minimal performance (throughput) that is dynamic over time. Throughput is selected because it adapts well to RUBiS and the workload. We can use other metrics like the response time, or more complex ones like "gold clients", clients who might make a purchase using the system if it is an e-business application. The SLA represents the number of request per second we should attempt to get. Our resource manager should be able to obtain (if possible) the required throughput using the minimal number of processors. The throughput obtained is drawn with a line, and we can see how it outlines the required SLA, however, there are some zones that we will explain further in a separate analysis done below. To finish we have a third plot with the CPU assignation for the two servers. We can see how they seem to follow the SLA in nearly all the tests, but there are two zones where the simulation introduced differences.

Although it is inside the same test we split the Globus result; it doesn't use a temporal axis and can be confusing. The results can be found in figure 5: boxes represent the response time SLA required for the almost 85 sessions, points represent successfuly finished sessions (we can see that we have 2 sessions that exceed the SLA with a difference of 2 seconds), and filled squares represent the same test without QoS Broker. We can see that, although we run more

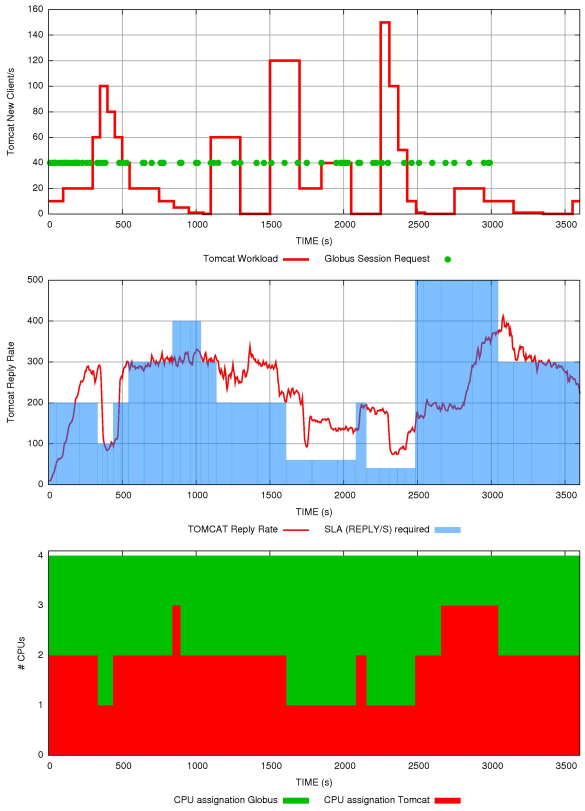

Figure 4. Evaluation of an heterogeneous environment using a resource manager using prediction.

sessions, we didn't fulfill the required SLA and the mean response time exceeded 500 seconds. Although the number of sessions accepted is low, we must note that the test is very intensive and is scaled for a larger number of processors. Long sessions complicate the task of the QoS Broker when the server capacity is heavily reduced. Finally, to analyze and explain the results we have selected a set of intervals from figure 4, where we can see interesting situations. It is noteworthy that, although the simulation and the manager are fully aware of the behavior of the workload (RUBiS), we can create a more comprehensive solution using learning techniques as we have done in the case of some experiments using Globus.

0-250 As we explained before the workload is additive, so at this point we have a workload of nearly 20 clients per second, but the system is new and requires a time to increase its workload (and thus its throughput). It requires a warm-up time to be stable.

400-500 Although the workload is high, the SLA requires a throughput of 100 replies per second. The resource

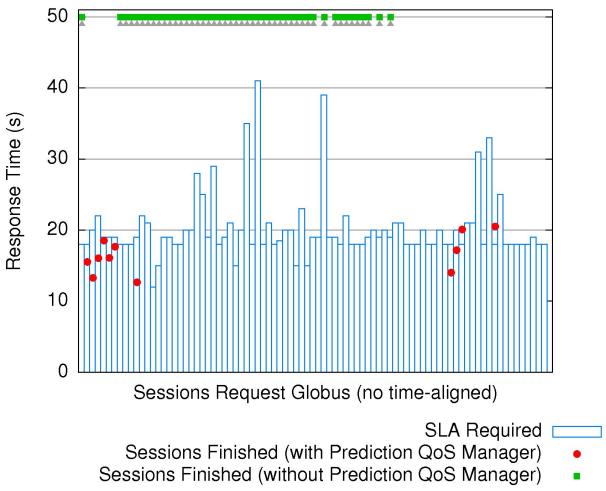

Figure 5. Globus Server evaluation in the shared server using the resource manager with prediction.

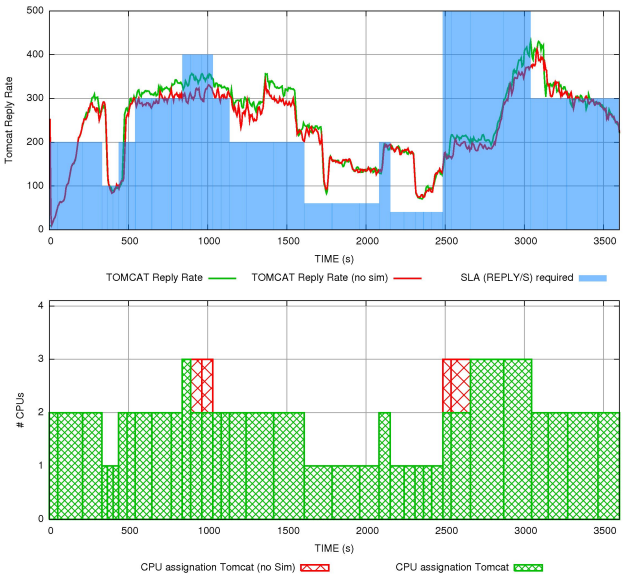

Figure 6. Using a resource manager with a static resource policy, we can see the difference in the assignation of processors without increasing the performance on the throughput.

manager decides, with the help of the simulator, that the number of processors required can be 1. Throughput falls to the specified value.

800-1000 In this interval, we can see an intelligent decision from the resource manager, where the SLA requires 400 replies per second. We can obtain this value with 3 processors, but after this decision we see that the workload decreases to a level that no longer requires all 3 so we decrease the assignation to 2 processors.

2500-3000 We have a similar behavior in this period: The SLA is high, but the workload is low so 2 processors are enough (1 will decrease performance). Eventually the workload increases, so we asked for 3 processors for Tomcat, increasing its performance to the required SLA value.

4.2 Resource Manager without Prediction

In figure 6, we compare the results with and without prediction. We have implemented a static policy. In the plots we can see that there are two zones ([800-1000] and [2500-3000]) where the approach using simulation is smarter because it can decide to use less processors. Those situations will increase as long as we increase the variables of the system (like using more processors or using smaller units for the resources). Simulation is more adaptable to these situations and can be done with lower effort.

Although it seems that it produces nearly the same results, we should note that the static table of decisions are found using simulation.

4.3 Simulation latency

The latency of the simulation is, as we have seen on previous work, low. We can simulate a scenario using one CPU in less than 10 seconds and then make a good decision. This time can be decreased further, and in most cases we will need no more than 5 seconds to make a decision. To reduce this time, we can add other techniques like the utilization of simulation cache to use old values and scenarios improving resource manager performance.

5 Conclusions

In this paper we demonstrate the application of simulation inside a resource manager. We use two simulations to decide how to dynamically manage the resources inside a shared server, and they provide us with more flexibility since we did not need to modify the application to inform us of its needs as was required in [15], where we implemented bidirectional communication between the heterogeneous middleware and the O.S. level to increase the performance of a shared server. We were able to obtain similar results using simulation and without the need to modify the middleware.

Sharing a server, with or without virtualization, for different applications is a solution to reduce power usage, and

increase the productivity of the servers in a data center. The best benefits are the ability to dynamically assign resources, but in those cases we possibly also need to inform the applications, or its managers, of the changes in the environment. A global resource manager, or VM resource manager, is required to do this. Simulation can provide a good way of predicting the behavior of the system, going farther than classic control theory. Simulation gives us more flexibility to deal with situations that would be difficult to specify with other methods. We have shown how we can maintain the performance and adapt the resources to the applications needs (Tomcat) and adapt the workload (Globus) to the resources. Finally, we should be aware that we are on a virtualized environment and apply or add this knowledge to the simulations.

On-line, real time simulation opens a number of promising possibilities in the creation of autonomic environments.

Acknowledgment

This work is supported by the Spanish Ministry of Science and Technology and the European Union under contract TIN2004-07739-C02-01 and TIN2007-60625. Thanks to Ferran Julià and Íñigo Goiri for their help.

References

[1] C. Amza, A. Chanda, E. Cecchet, A. Cox, S. Elnikety, R. Gil, J. Marguerite, K. Rajamani, and W. Zwaenepoel. Specification and implementation of dynamic web site benchmarks. *Specification and Implementation of Dynamic Web Site Benchmarks, in Fifth Annual IEEE International Workshop on Workload Characterization (WWC-5)*, 2002.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM.

[3] R. Buyya and M. Murshed. GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *The Journal of Concurrency and Computation: Practice and Experience (CCPE), Wiley Press*, 14 (issue 13-15), Nov-Dec 2002.

[4] A. Dan, H. Ludwig, and G. Pacifici. Web Service Differentiation with Service Level Agreements. In *White Paper, IBM Corporation*, 2003.

[5] B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham, and R. Neugebauer. Xen and the Art of Virtualization. In *Proceedings of the ACM Symposium on Operating Systems Principles*, 2003.

[6] I. T. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. In *Proceedings of the 2005 IFIP International Conference on Network and Parallel Computing*, pages 2–13, 2005.

[7] A. Karve, T. Kimbrel, G. Pacifici, M. Spreitzer, M. Steinder, M. Sviridenko, and A. Tantawi. Dynamic placement for clustered web applications. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 595–604, New York, NY, USA, 2006. ACM.

[8] S. Kounev and A. Buchmann. Performance modelling of distributed e-business applications using queuing petri nets. In *Proc. of the 2003 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'03)*, Mar. 2003.

[9] S. Kounev and A. Buchmann. SimQPN - a tool and methodology for analyzing queueing Petri net models by means of simulation. *Performance Evaluation*, 63(4-5):364–394, May 2006.

[10] S. Kounev, R. Nou, and J. Torres. Autonomic QoS-aware resource management in grid computing using online performance models. *Second International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS-2007), Nantes, France*, October 23-25 2007.

[11] W. Kreutzer, J. Hopkins, and M. van Mierlo. Simjava: a framework for modeling queueing networks in java. In *WSC '97: Proceedings of the 29th conference on Winter simulation*, pages 483–488, Washington, DC, USA, 1997. IEEE Computer Society.

[12] D. Mosberger and T. Jin. httperf: A tool for measuring web server performance. *WISP'98,Madison, Wisconsin, USA*, 42 (2-3):59–67, June 23 1998.

[13] M. J. North, N. T. Collier, and J. R. Vos. Experiences creating three implementations of the repast agent modeling toolkit. *ACM Trans. Model. Comput. Simul.*, 16(1):1–25, 2006.

[14] R. Nou, J. Guitart, and J. Torres. Simulating and modeling secure web applications. In V. N. Alexandrov, G. D. van Albada, P. M. A. Sloot, and J. Dongarra, editors, *International Conference on Computational Science (1)*, volume 3991 of *Lecture Notes in Computer Science*, pages 84–91. Springer, 2006.

[15] R. Nou, F. Julià, J. Guitart, and J. Torres. Dynamic resource provisioning for self-adaptive heterogeneous workloads in smp hosting platforms. *International Conference on E-business (2nd) ICE-B 2007, Barcelona, Spain.*, July 28th-31th 2007.

[16] R. Nou, S. Kounev, F. Julia, and J. Torres. Autonomic QoS control in enterprise Grid environments using online simulation. *Journal of Systems and Software*, to appear, 2008.

[17] R. Nou, S. Kounev, and J. Torres. Building online performance models of grid middleware with fine-grained load-balancing: A globus toolkit case study. In K. Wolter, editor, *EPEW*, volume 4748 of *Lecture Notes in Computer Science*, pages 125–140. Springer, 2007.

[18] P. Padala, K. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem. Adaptive Control of Virtualized Resources in Utility Computing Environments. In *Proceedings of the 2007 conference on EuroSys*, 2007.

[19] A. Varga. The OMNeT++ discrete event simulation system. In *European Simulation Multiconference (ESM'2001)*, June.